# Design Document

for

# FinXpert

Version <1.0>

Prepared by


**Group 16:** **Group Name:** Tech Titans

| | | |
|---|---|---|
| Dwij Om Oshoin | 220386 | dwij22@iitk.ac.in |
| Karthik S Pillai | 220502 | karthiksp22@iitk.ac.in |
| Sumit Kumar | 221102 | sumitkumar22@iitk.ac.in |
| Aniket Kumar Choudhary | 220140 | aniketkc22@iitk.ac.in |
| Devank Saran Prajapati | 220339 | devanks22@iitk.ac.in |
| Rahul Mahato | 220859 | mrahul22@iitk.ac.in |
| Ayush Yadav | 220267 | aarchit22@iitk.ac.in |
| Archit Atrey | 220195 | ayushku22@iitk.ac.in |
| Madhav Khetan | 220596 | madhavk22@iitk.ac.in |
| Piyush Meena | 220768 | piyushm22@iitk.ac.in |


**Course:** CS253

**Mentor TA:** Naman Baranwal

**Date:** 07/02/2025

# Revisions

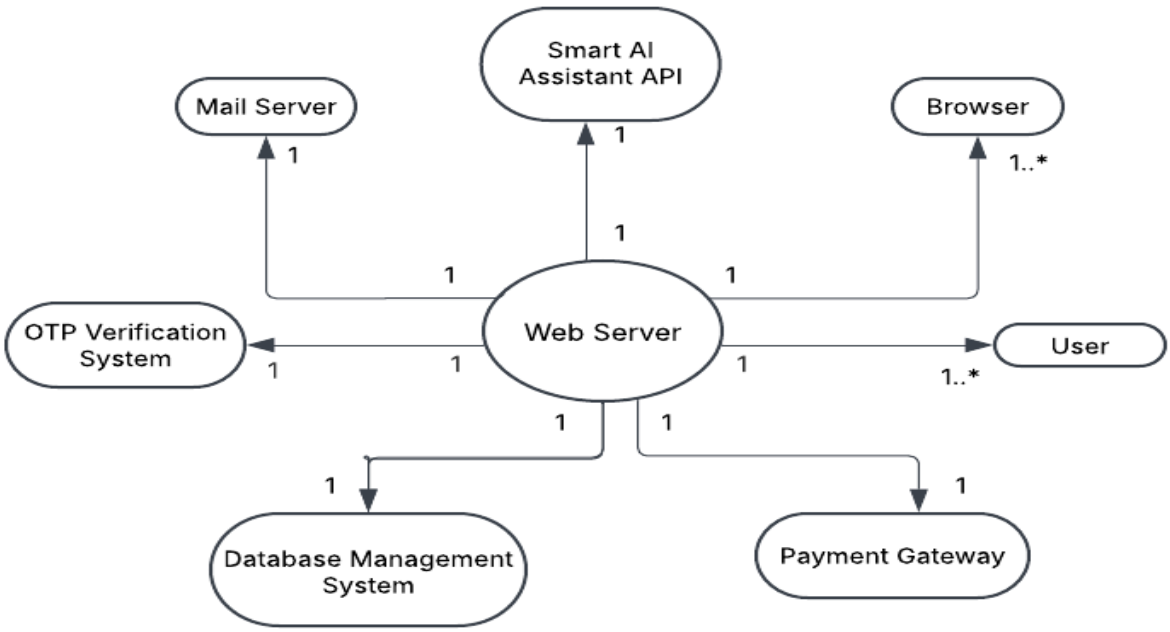
| Version | Primary Author(s) | Description of Version | Date Completed |
|---|---|---|---|
| 1.0 | Dwij Om Oshoin<br>Karthik S Pillai<br>Sumit Kumar<br>Aniket Kumar Choudhary<br>Devank Saran Prajapati<br>Rahul Mahato<br>Ayush Yadav<br>Archit Atrey<br>Madhav Khetan<br>Piyush Meena | Completed first version of the document | 07/02/25 |

# 1 Context Design

## 1.1 Context Model

The context model shows the various entities that interact with the system, and the relationships between them. It plays a crucial role in supporting efficient context management. A brief explanation of the context model is as follows -

- **Smart AI Assistant API** – The core intelligence that provides users with financial insights, expense tracking, and budgeting recommendations. It processes user queries, fetches data from the database, and interacts with external services like the payment gateway for transaction analysis.
- **Browser** – The user interface for interacting with the AI assistant, viewing financial insights, making transactions, and managing personal expenses.
- **Mail Server** – Handles email communications, including transaction alerts, financial summaries, authentication emails, and user notifications.
- **OTP Verification System** – Ensures secure user authentication for accessing financial data and authorizing transactions by generating and verifying one-time passwords.
- **Payment Gateway** – Facilitates secure financial transactions, allowing users to track expenses, make payments, and analyze spending patterns through the AI assistant.
- **User** – The end-user who interacts with the system for financial planning, making payments, and receiving insights into spending habits.
- **DBMS (Database Management System)** – Stores and manages financial data, user preferences, transaction history, and AI-generated insights for personalized recommendations.

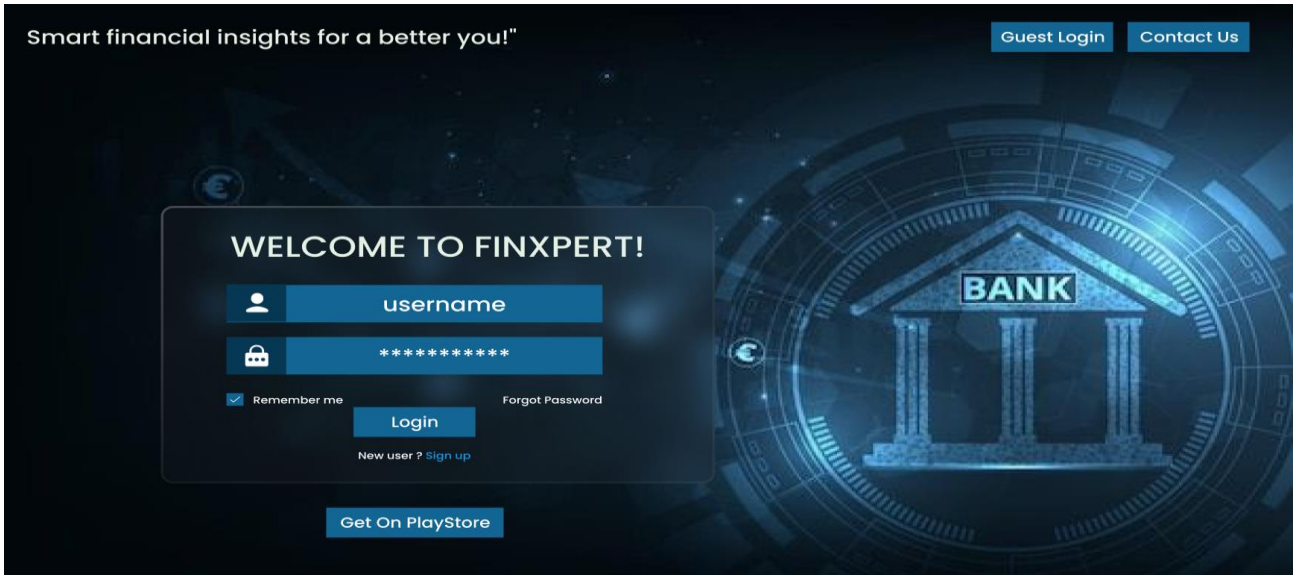


## 1.2 Human Interface Design

Users will engage with the system via a web-based platform featuring an intuitive user interface (UI) designed for seamless navigation. In the future, we plan to develop a mobile application version to enhance accessibility across devices.

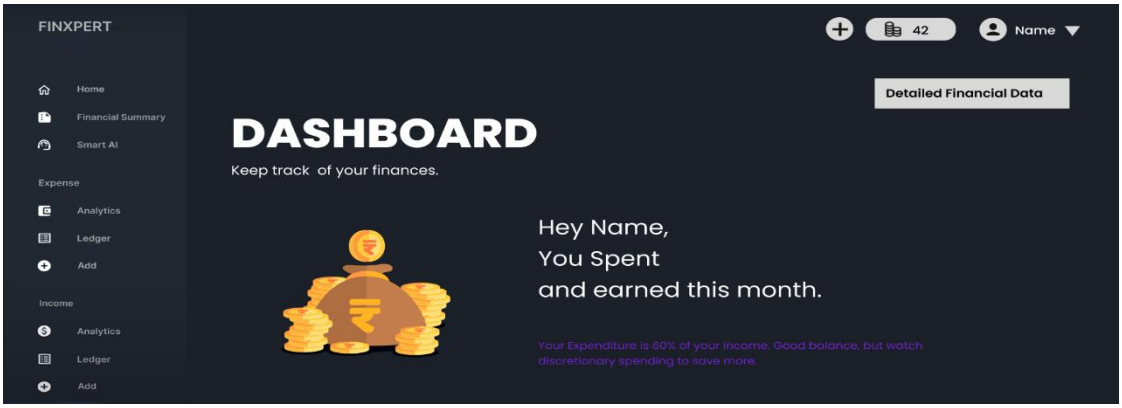
### 1.2.1 Login Page

To access the application, the user must log in first. If the user does not have an account, they can sign up for a new one. In case of a forgotten password, the "Forgot Password" option allows for easy reset.



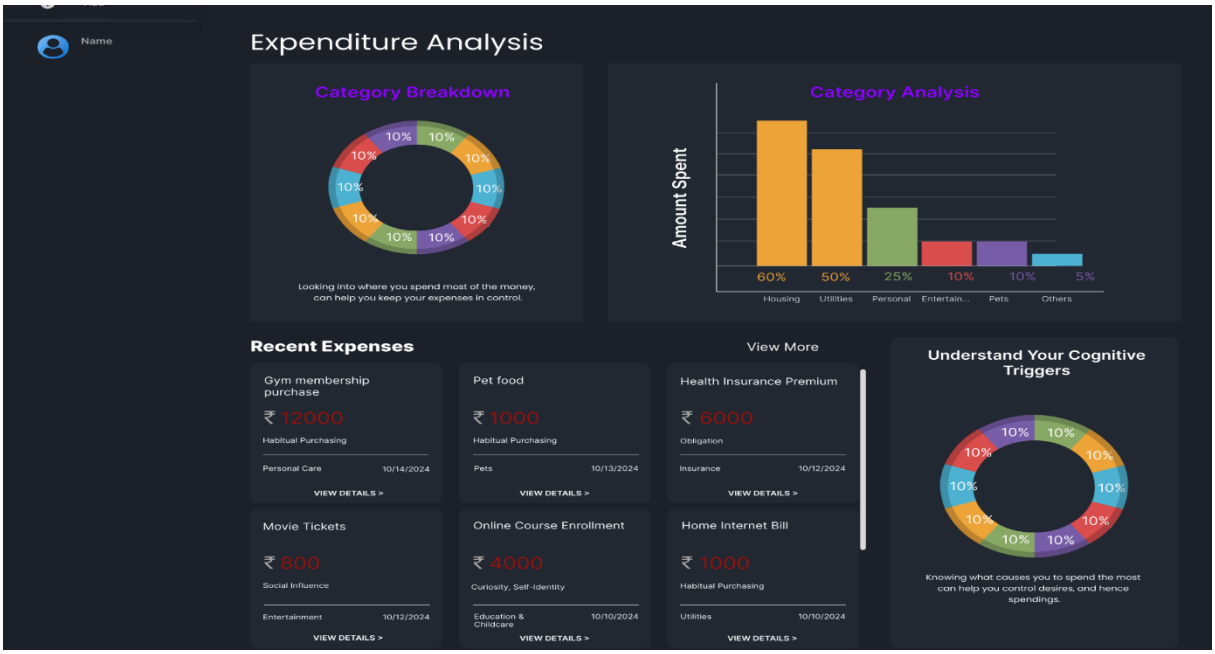

### 1.2.2. Dashboard

The Dashboard Page provides a quick overview of the user's financial health by displaying current month expenses and earnings in a clear and concise manner. It analyzes the expenditure-to-earning ratio and offers insights into whether the user's spending habits are sustainable for a secure financial future. Based on this analysis, the page provides brief, actionable advice to help users maintain a balanced financial portfolio, optimize savings, and make informed financial decisions.



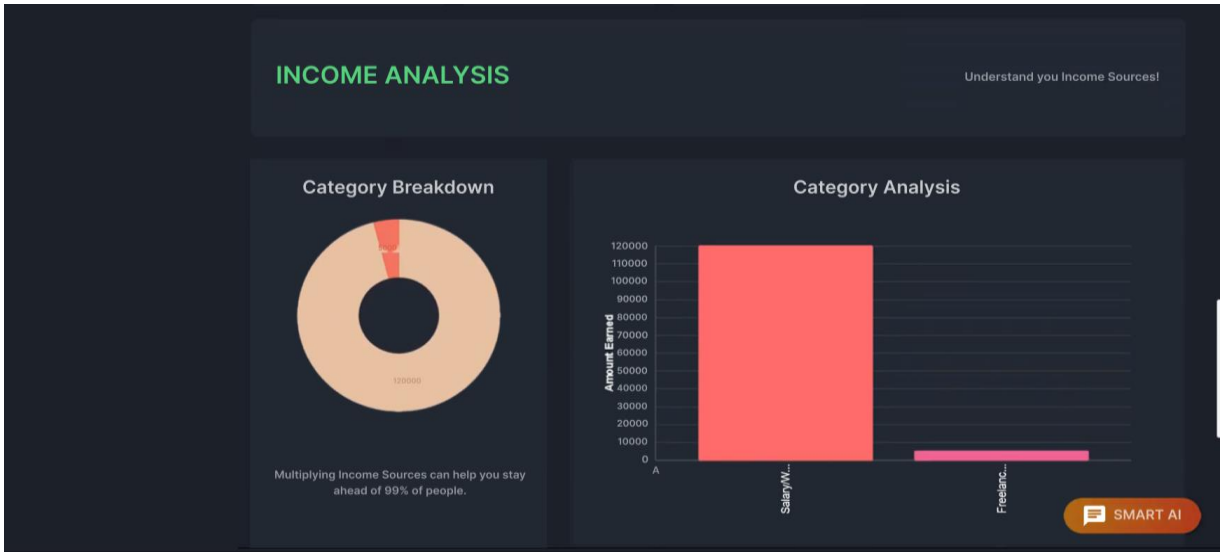
### 1.2.3 Expenditure Analysis

It shows a pie chart showing category breakdowns, a bar chart for category analysis, recent expenses list, and a pie chart explaining cognitive triggers in spending habits.



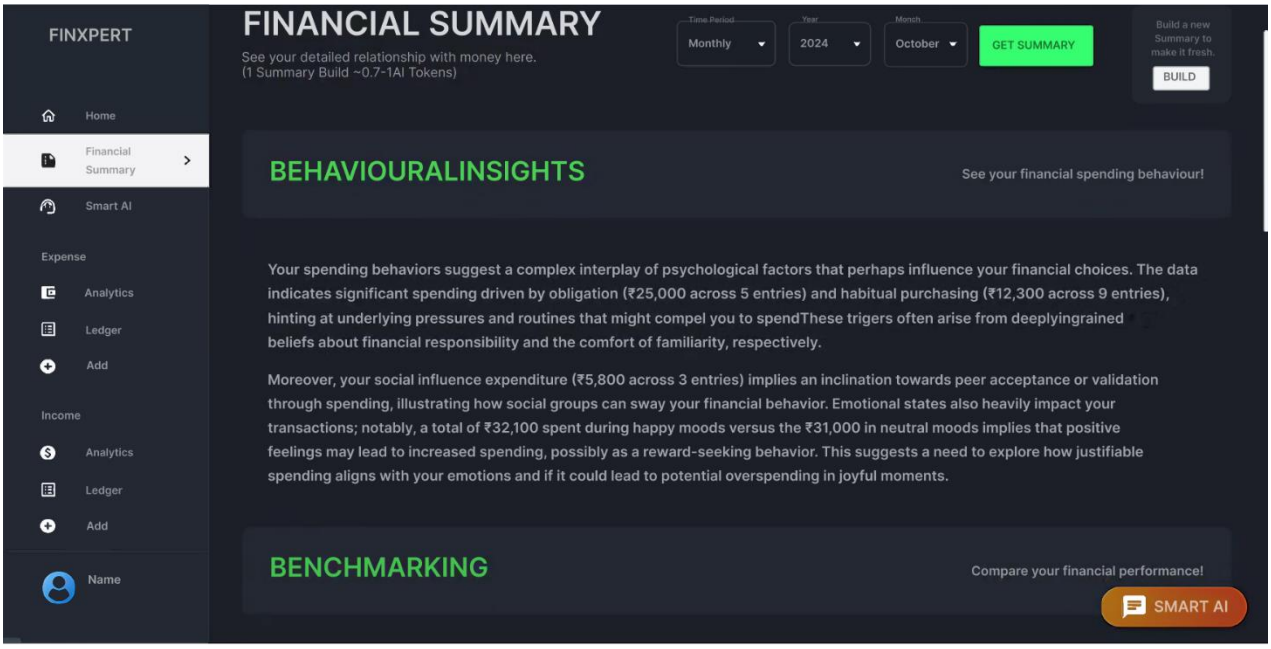

### 1.2.4 Income Analysis

The Income Analysis page provides users with a detailed breakdown of their earnings, categorizing income sources such as salary, freelance work, and investments. It visualizes income trends over time, offering insights into growth patterns and fluctuations. Users can track their progress toward financial goals, analyze income distribution, and receive AI-driven forecasts for future earnings.



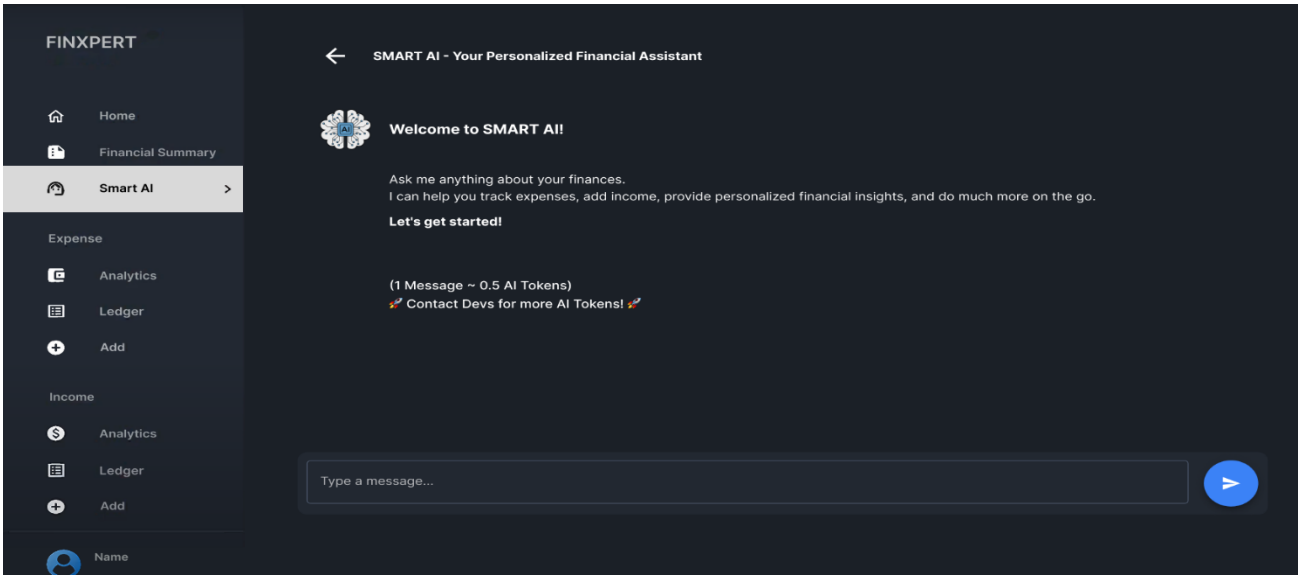

### 1.2.5 Financial Summary

This page gives a rough summary of user's expenditure pattern based on the behavioural insights and other factors like social influence.



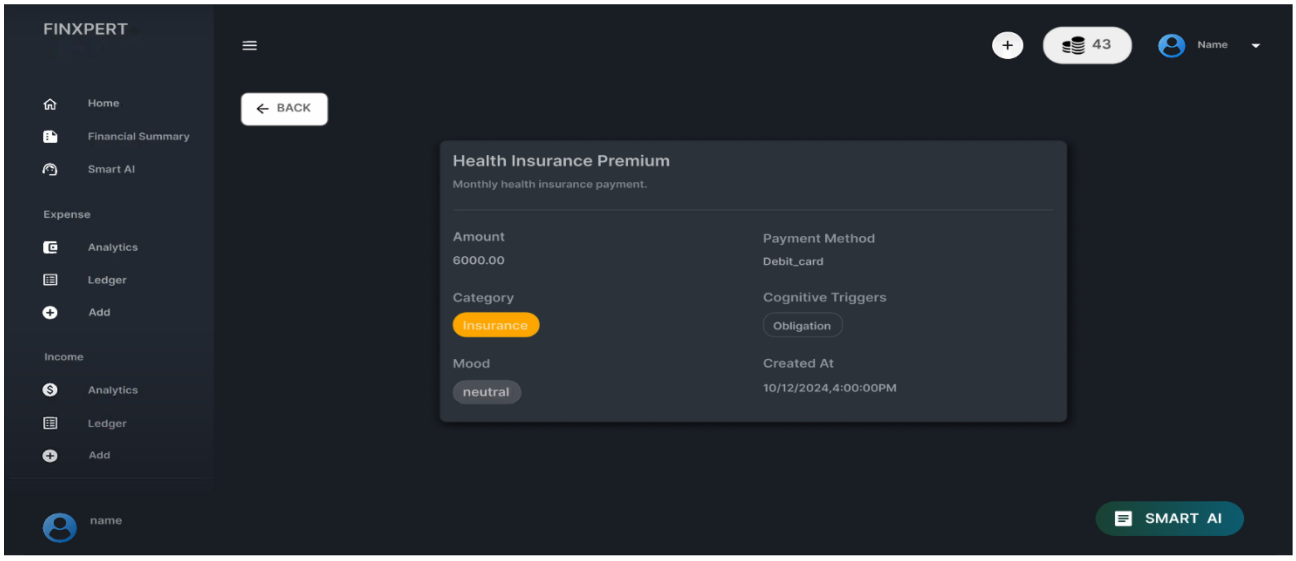

### 1.2.6 Smart AI

The Smart AI is the personal assistant that provides users with personalized financial insights, expense tracking and budgeting recommendations. Additionally, it also assists with user queries and data fetching.



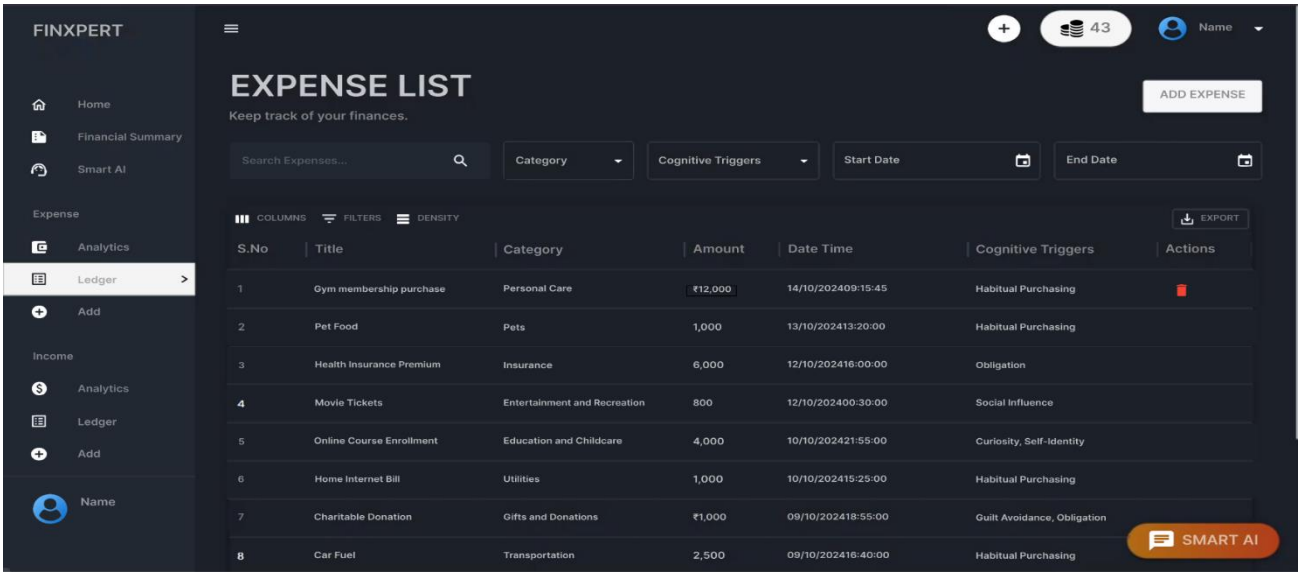

### 1.2.7 View All

The "View All" feature offers detailed insights into a selected expense, displaying its title, amount, category, payment method, mood, cognitive triggers, and creation date. It provides a clear snapshot for easy tracking and analysis.



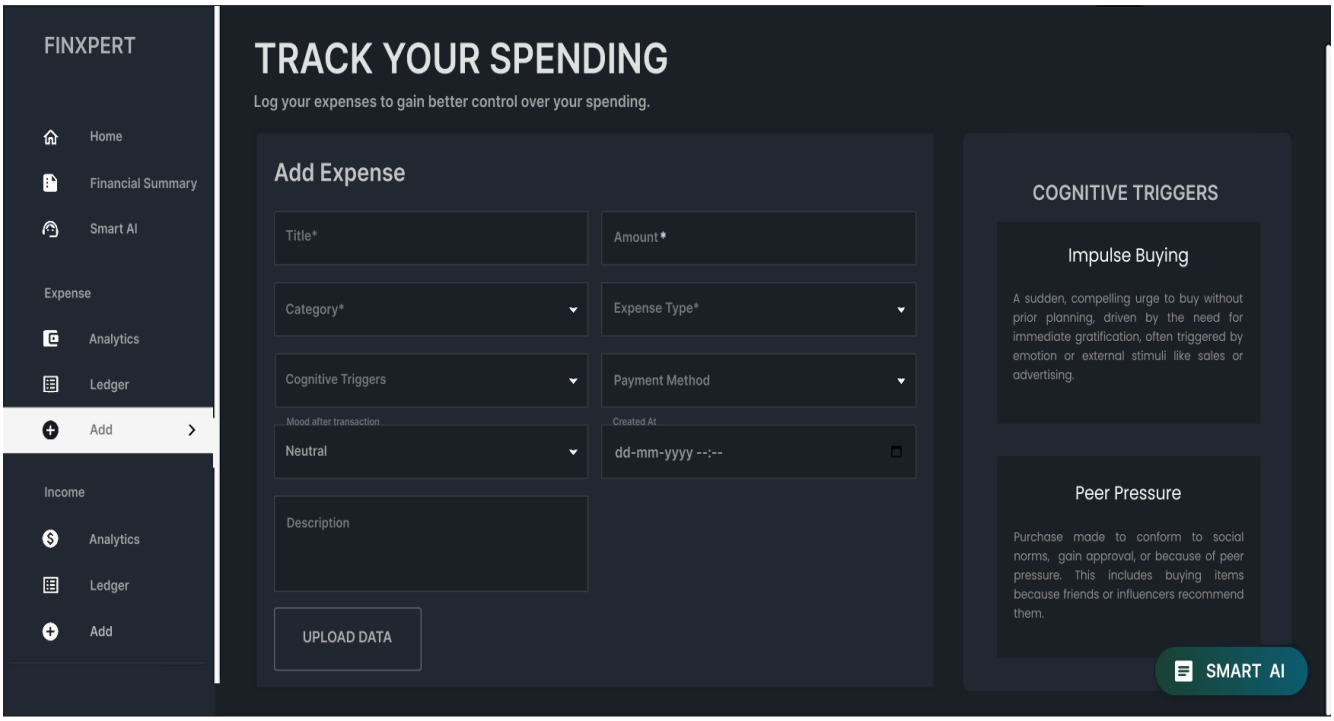

### 1.2.8 Ledger

The "Ledger" section provides a comprehensive overview of all recorded expenses in a tabular format. Users can view key details such as the title, category, amount, date and time, and cognitive triggers associated with each expense. The interface includes options to filter and sort entries by various parameters like category, cognitive triggers, or date range. Additionally, users can export the ledger for external use or delete specific records for better management. This feature ensures users can track and analyze their spending patterns.

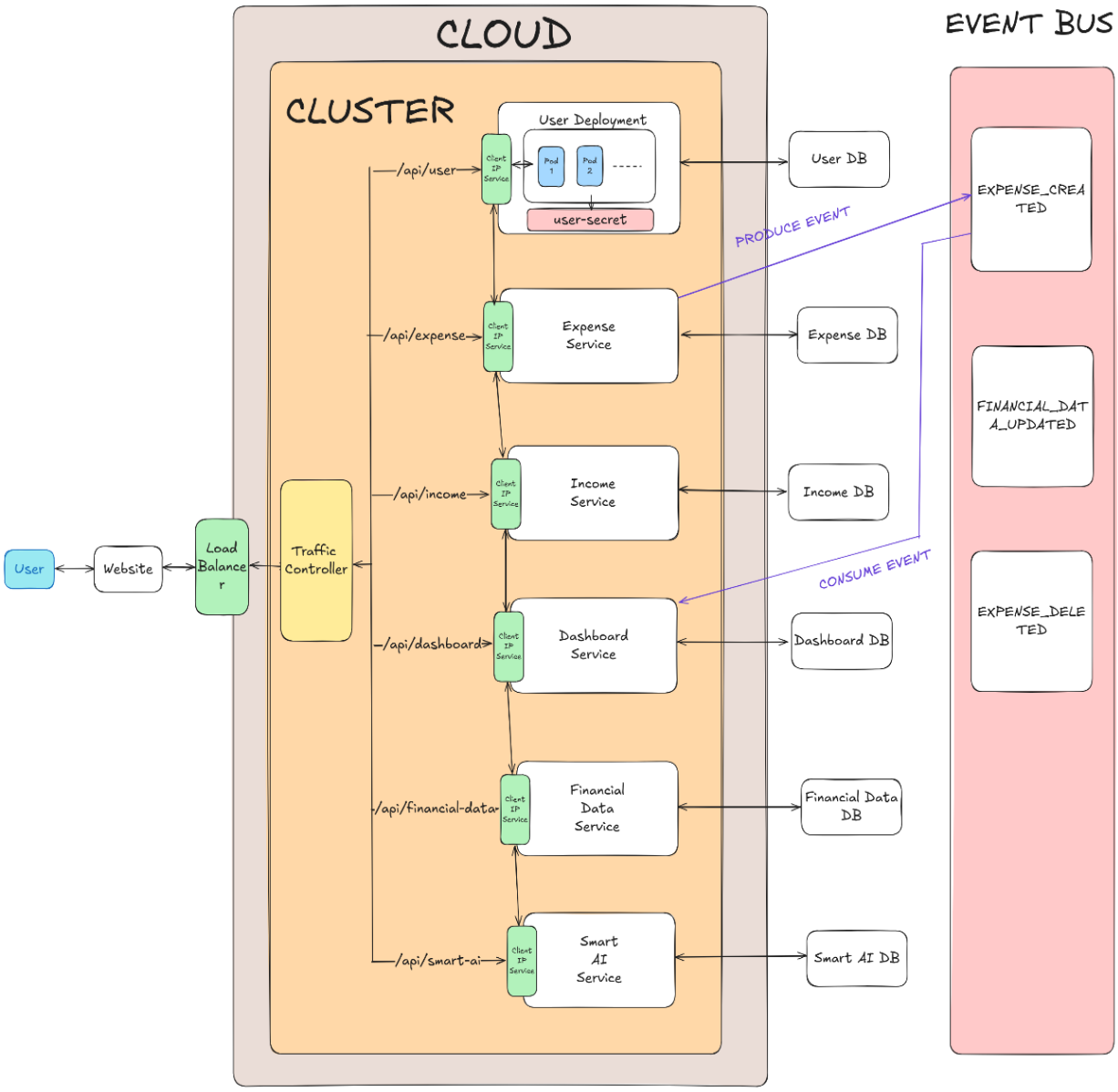



### 1.2.9 Add

The "Add Expense" section provides users with a detailed form to log their expenses efficiently. It includes fields such as title, amount, category, and expense type. Additionally, users can select relevant "Cognitive Triggers" from the panel on the right, like impulse buying or peer pressure, to better understand the behavioral factors influencing their spending. This feature aims to promote awareness and control over financial habits.


FINXPERT
Home
Financial Summary
Smart AI
Expense
Analytics
Ledger
Add
Income
Analytics
Ledger
Add
TRACK YOUR SPENDING
Log your expenses to gain better control over your spending.
Add Expense
Title*
Amount*
Category*
Expense Type*
Cognitive Triggers
Payment Method
Mood after transaction
Neutral
Created At
dd-mm-yyyy --:--
Description
UPLOAD DATA
COGNITIVE TRIGGERS
Impulse Buying
A sudden, compelling urge to buy without prior planning, driven by the need for immediate gratification, often triggered by emotion or external stimuli like sales or advertising.
Peer Pressure
Purchase made to conform to social norms, gain approval, or because of peer pressure. This includes buying items because friends or influencers recommend them.
SMART AI

# 2 Architecture Design



# Key Architectural Components

## Microservices Architecture

FinXpert follows a microservices-based architecture where each service is designed to handle a specific functionality, ensuring modularity and ease of maintenance.

1. **User Service**
    a. Manages user authentication, authorization, and profile management.
    b. Uses PostgreSQL (Alven) for relational data storage such as user credentials.
    c. Implements Kubernetes Secrets to manage sensitive authentication data securely.
2. **Expense Service**
    a. Handles expense tracking, categorization, and management.
    b. Stores unstructured expense data in MongoDB (Atlas).
    c. Produces events to Kafka when a new expense is created (e.g., EXPENSE_CREATED).
3. **Income Service**
    a. Manages income entries, sources, and associated details.
    b. Uses MongoDB (Atlas) for flexible income data storage.
    c. Publishes events to Kafka for income updates and changes.
4. **Financial Data Service**
    a. Analyzes and processes financial data for insights and summaries.
    b. Consumes financial data events from Kafka (e.g., FINANCIAL_DATA_UPDATED).
    c. Stores financial data in MongoDB (Atlas) for analytics and forecasting.
5. **Dashboard Service**
    a. Aggregates frequently accessed financial data for fast retrieval and display.
    b. Enhances user experience by reducing API latency through caching mechanisms.
    c. Uses MongoDB (Atlas) as the backend storage.
6. **Smart AI Service**
    a. Provides personalized financial insights using AI/ML techniques.
    b. Uses MongoDB (Atlas) to store AI-generated cognitive triggers and insights.
    c. Integrates with Kafka for event-driven data updates.

## 2.1 Event-Driven Communication: Kafka Event Bus

Kafka is employed as the central event bus to enable asynchronous, decoupled inter-service communication. It manages real-time financial event processing with dedicated topics for different types of events:

- EXPENSE_CREATED: Triggered when a user logs a new expense.
- FINANCIAL_DATA_UPDATED: Emitted when financial summaries are recalculated.
- EXPENSE_DELETED: Occurs when a user deletes an expense entry.
- Other event types as needed for real-time updates across services.

Zookeeper is used for managing Kafka clusters and ensuring distributed coordination.

## 2.2 Cloud Infrastructure & Orchestration

### 2.2.1 Google Cloud Platform (GCP) Services

- Compute: Kubernetes Cluster deployed on GCP for managing containerized services.
- Networking: Load Balancer (GCP) ensures optimal traffic distribution.
- Storage: Cloud-hosted databases (PostgreSQL for user data, MongoDB for financial data).

### 2.2.2 Kubernetes & Deployment Strategy

- Ingress Controller (NGINX): Routes incoming traffic to the appropriate microservices.
- Pods & Scaling: Each microservice is deployed as a separate pod, with autoscaling enabled to handle varying workloads.
- Cron Jobs: Used for scheduled tasks like guest token resets and automated financial data analysis.

## 2.3 Security & Compliance

Security is a critical aspect of FinXpert 's architecture. The following measures are in place to ensure data protection and compliance:

- Kubernetes Secrets: Securely manages sensitive data such as API keys, database credentials, and authentication secrets.
- TLS/SSL Encryption: Ensures secure communication between services and user interactions.
- Role-Based Access Control (RBAC): Enforces user permissions and service-to-service authentication.

## 2.4 User Interaction Flow

1. The user accesses FinXpert via the Web Application (React). (Mobile app is currently unavailable.)
2. The request passes through the GCP Load Balancer and is routed by the Ingress Controller (NGINX).
3. The relevant microservice (User, Expense, Income, etc.) processes the request.
4. If necessary, the service interacts with the database (PostgreSQL for user data, MongoDB for financial data).
5. Events are published to Kafka for real-time updates across other services.
6. The Dashboard Service aggregates and presents the processed data back to the user.

## 2.5 Conclusion

FINXPERT's architecture is designed for robustness, scalability, and efficiency. By leveraging Kubernetes for orchestration, Kafka for event-driven communication, and a combination of PostgreSQL and MongoDB for data storage, FINXPERT ensures a high-performance and secure financial management system. The modular microservices architecture enables easy maintenance and future enhancements, making it a future-ready solution for personal finance management.

# 3 Object Oriented Design

## 3.1 Use Case Diagrams

### Use Case 1: User Registration and Authentication


**Purpose: Allow users to create an account and authenticate using secure OTP-based verification.**

**Requirements Traceability:**
- **F1:** User Account Management – Enable secure signup and OTP authentication.
- **F8:** Security and Privacy – Encrypt user data and prevent abuse with rate limiting.

**Priority:** High

**Preconditions:**
- User must provide a valid email.
- Email service and OTP service must be operational.

**Postconditions:**
- The user account is created and authenticated.
- User session is initiated upon successful authentication.

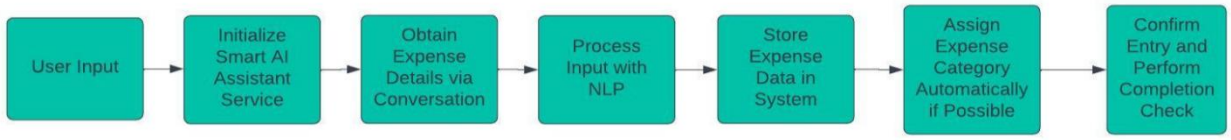
**Actors:**
- User
- Email Service
- OTP Service

**Exceptions:**
- OTP delivery failures due to network issues.
- Account creation errors if the email is already registered.

**Includes:**
- None.

**Notes/Issues:**
- Consider adding CAPTCHA to prevent automated abuse.

## Use Case 2: Adding an Expense via the Smart AI Assistant




**Purpose:** Enable users to add expense entries using conversational input through the Smart AI Assistant.

**Requirements Traceability:**

- **F2:** Expense Management – Support expense CRUD operations.
- **F6:** Smart AI Assistant – Process conversational input for adding expenses.

**Priority:** High

**Preconditions:**

- User must be authenticated.
- Smart AI Assistant service must be functional.

**Postconditions:**

- Expense data is added to the system.
- Expense category is auto-assigned if possible.

**Actors:**

- User
- Smart AI Assistant

**Exceptions:**

- Misunderstanding of user input by the NLP system.
- Backend service unavailability for storing the expense.

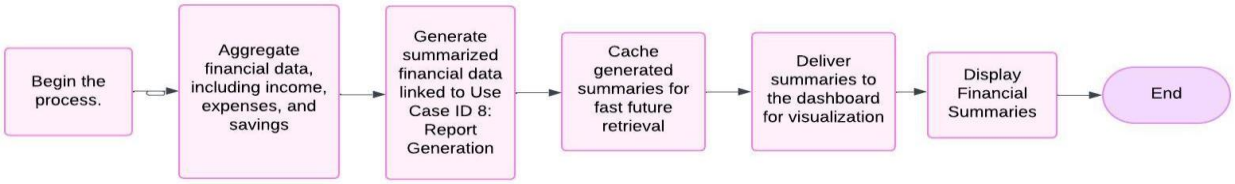
**Includes:**

- Use Case ID 6: Expense Categorization.

**Notes/Issues:**

- Continuously train the NLP model for better accuracy in understanding user queries.

## Use Case 3: Generating Financial Summaries




**Purpose:** Provide users with quick financial summaries, including total income, expenses, and savings.

**Requirements Traceability:**

- **F4:** Financial Data Analysis – Aggregate data for insights.
- **F5:** Dashboard Functionality – Deliver summaries to the dashboard for instant retrieval.

**Priority:** Medium

**Preconditions:**

- User must have sufficient financial data in the system.
- The system must have access to up-to-date financial records.

**Postconditions:**

- A summarized view of financial data is displayed on the dashboard.
- Summaries are cached for faster retrieval.

**Actors:**

- User
- Dashboard Service

**Exceptions:**

- Data aggregation errors due to inconsistent data sources.
- Dashboard visualization failures due to service outages.

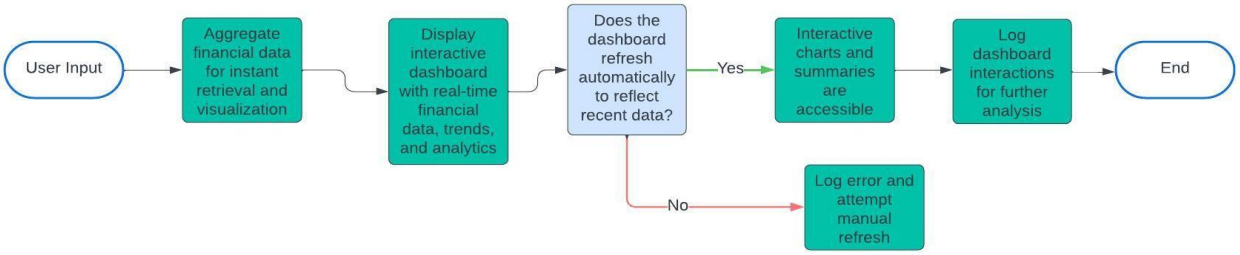
**Includes:**

- Use Case ID 8: Report Generation.

**Notes/Issues:**

- Optimize aggregation for high volumes of financial data.

## Use Case 4: Dashboard Visualization




**Purpose:** Display real-time financial data, trends, and analytics in an interactive and user-friendly dashboard.

**Requirements Traceability:**
- **F5:** Dashboard Functionality – Aggregate financial data for instant retrieval and visualization.
- Requirement ID 10: Dashboard must refresh automatically to reflect recent data.
- Requirement ID 11: Interactive charts and summaries should be accessible.

**Priority:** High

**Preconditions:**
- User must be logged in.
- Dashboard Service must have access to aggregated financial data.

**Postconditions:**
- Users can view up-to-date financial analytics.
- Dashboard interactions are logged for further analysis.

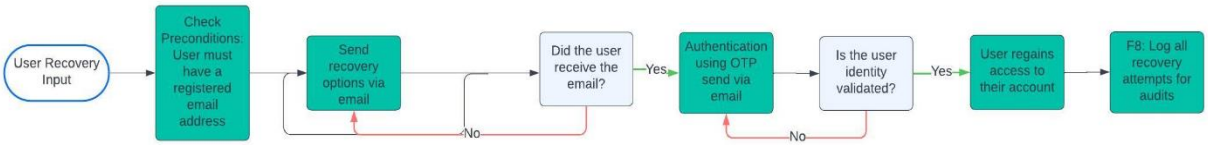
**Actors:**
- User
- Dashboard Service

**Exceptions:**
- Failure to fetch updated data from the backend.
- High latency in rendering visualizations.

**Includes:**
- Use Case ID 3: Generating Financial Summaries.

**Notes/Issues:**
- Optimize the dashboard for mobile responsiveness.
- Test scalability for large datasets to maintain performance.

## Use Case 5: Secure Account Recovery Mechanism




**Purpose:** Allow users to recover their accounts in case of forgotten credentials or account lockouts.

**Requirements Traceability:**

- **F1:** User Account Management – Enable secure recovery mechanisms.
- **F8:** Security and Privacy – Encrypt data and log recovery attempts for audits.
- Requirement ID 12: Users must receive recovery options via email.
- Requirement ID 13: Secure authentication must validate user identity during recovery.

**Priority:** Medium

**Preconditions:**

- User must have a registered email address.
- The OTP generation service must be functional.

**Postconditions:**

- The user regains access to their account.
- All recovery attempts are logged for security audits.

**Actors:**

- User
- OTP generation service
- Email service provider

**Exceptions:**

- Incorrect answers to security questions (if implemented).
- Delivery issues with recovery emails.

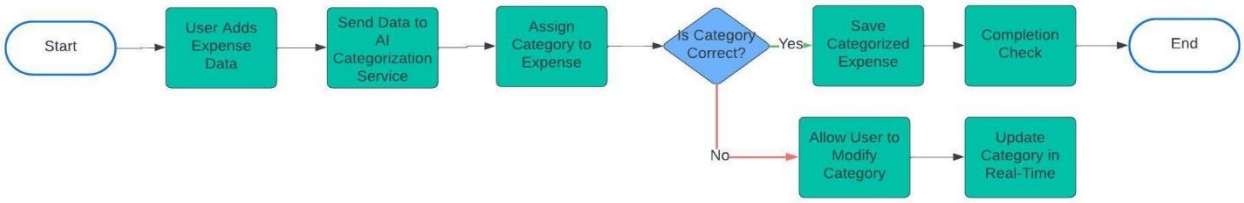
**Includes:**

- Use Case ID 1: User Registration and Authentication.

**Notes/Issues:**

- Evaluate the need for multi-factor authentication during recovery.
- Consider rate limiting recovery attempts to prevent abuse.

## Use Case 6: Expense Categorization




**Purpose:** Automatically categorize expenses based on user input and AI-generated insights.

**Requirements Traceability:**

- **F2:** Expense Management – Allow CRUD operations for expense entries.
- **F4:** Financial Data Analysis – Analyze spending habits and provide insights for better categorization.

**Priority:** High

**Preconditions:**

- Expense data must be added to the system.
- AI categorization service must be functional.

**Postconditions:**

- The expense is assigned an appropriate category.
- Category modifications, if any, are updated in real time.

**Actors:**

- User
- AI Categorization Service

**Exceptions:**

- Misclassification of expenses due to insufficient data.
- Category modification failure due to service unavailability.

**Includes:**

- Use Case ID 2: Adding an Expense via the Smart AI Assistant.

**Notes/Issues:**

- Implement a feedback loop to improve categorization accuracy over time.

## Use Case 7: Budget Tracking and Alerts


**Purpose:** Enable users to set budgets and receive alerts when nearing or exceeding budget limits.

**Requirements Traceability:**

- **F4:** Financial Data Analysis – Aggregate and analyze spending habits.
- **F5:** Dashboard Functionality – Provide real-time visualizations for budget tracking.

**Priority:** Medium

**Preconditions:**

- Users must define a budget.
- The system must have access to real-time expense data.

**Postconditions:**

- Users are notified of budget status via in-app alerts or email.
- Budget data is logged for analysis and reporting.

**Actors:**

- User
- Notification Service

**Exceptions:**

- Delay in real-time budget alerts due to system lag.
- Incorrect budget calculation caused by unprocessed expense data.

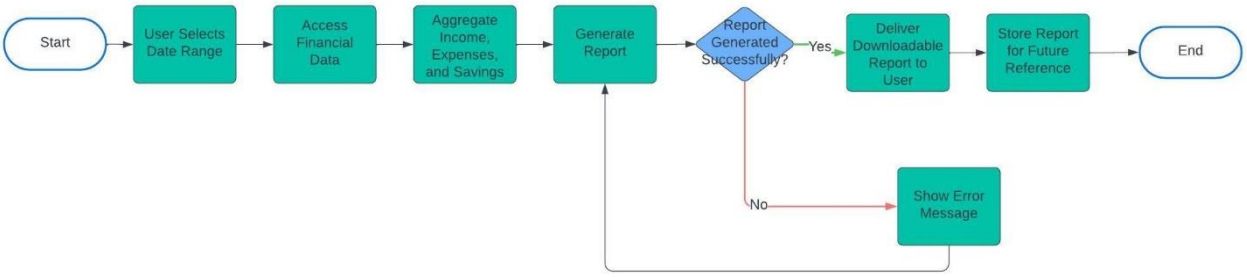
**Includes:**

- Use Case ID 6: Expense Categorization.

**Notes/Issues:**

- Add a feature for predictive alerts based on spending trends.

## Use Case 8: Report Generation




**Purpose:** Generate detailed financial reports for a selected period, including income, expenses, and savings.

**Requirements Traceability:**

- **F4:** Financial Data Analysis – Aggregate expense and income data for insights.
- **F5:** Dashboard Functionality – Deliver aggregated data to the dashboard for visualization.

**Priority:** Medium

**Preconditions:**

- User must select a date range for the report.
- Report generation service must have access to financial data.

**Postconditions:**

- The user receives a downloadable report.
- The report is stored for future reference in the user's account.

**Actors:**

- User
- Report on Generation Service

**Exceptions:**

- Incomplete report generation due to backend service failure.
- Export functionality errors for specific formats.

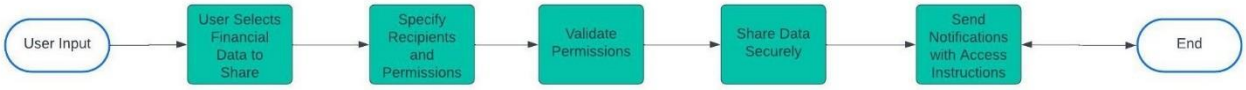
**Includes:**

- Use Case ID 3: Generating Financial Summaries.

**Notes/Issues:**

- Implement scheduled report generation for automated delivery.

## Use Case 9: Sharing Financial Insights




**Purpose:** Allow users to share specific financial insights or reports with others via email or social media.

**Requirements Traceability:**

- **F5:** Dashboard Functionality – Aggregate and share financial data.
- **F8:** Security and Privacy – Ensure secure sharing and restricted access.

**Priority:** Low

**Preconditions:**

- The user must specify recipients and permissions.
- Sharing service must validate recipient details.

**Postconditions:**

- Selected data is securely shared with recipients.
- Recipients receive notifications with access instructions.

**Actors:**

- User
- Sharing Service
- Notification Service

**Exceptions:**

- Sharing failure due to incorrect recipient details.
- Unauthorized access due to improper permission settings.

**Includes:**

- Use Case ID 8: Report Generation.

**Notes/Issues:**

- Consider introducing watermarking for sensitive shared reports.

## Use Case 10: Income Management

**Purpose:** Allow users to add and manage income sources and track income history.

**Requirements Traceability:**

- **F3:** Income Management – Allow CRUD operations for income entries.
- **F4:** Financial Data Analysis – Aggregate income data for financial insights.

**Priority:** High

**Preconditions:**

- User must be logged in.
- The system must have an operational income tracking module.

**Postconditions:**

- Income data is updated and displayed accurately in the dashboard.
- Changes are logged for future reference.

**Actors:**

- User
- Income Management Service

**Exceptions:**

- Failure to save new or updated income data due to backend issues.
- Discrepancies between income and expense data synchronization.

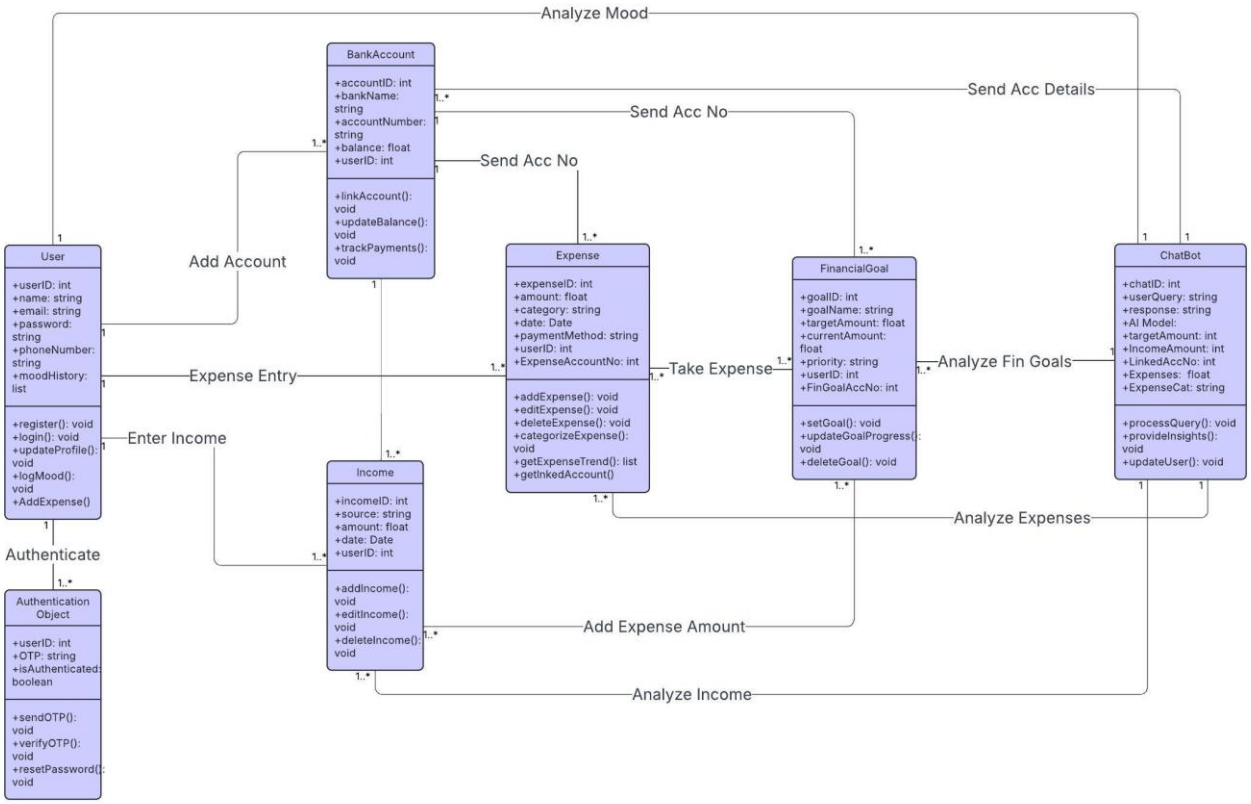
**Includes:**

- Use Case ID 3: Generating Financial Summaries.

**Notes/Issues:**

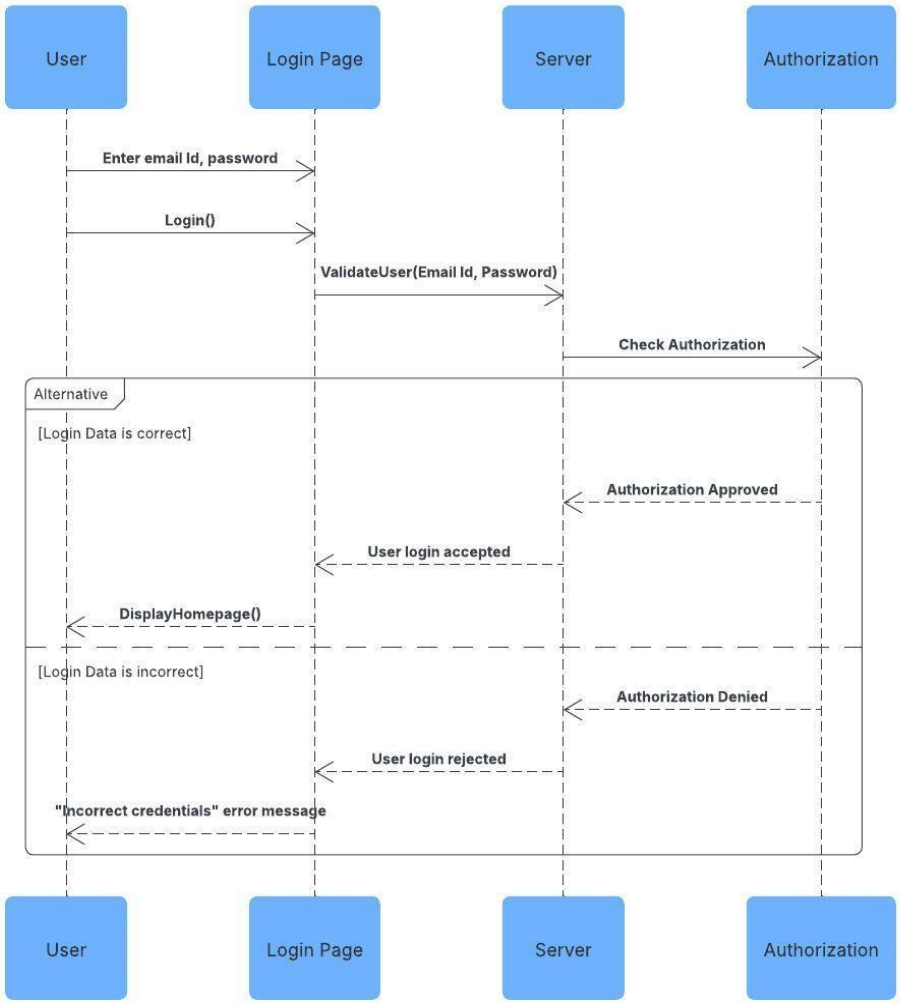
- Include integration with external payroll systems for automated updates.

## 3.2 Class Diagrams

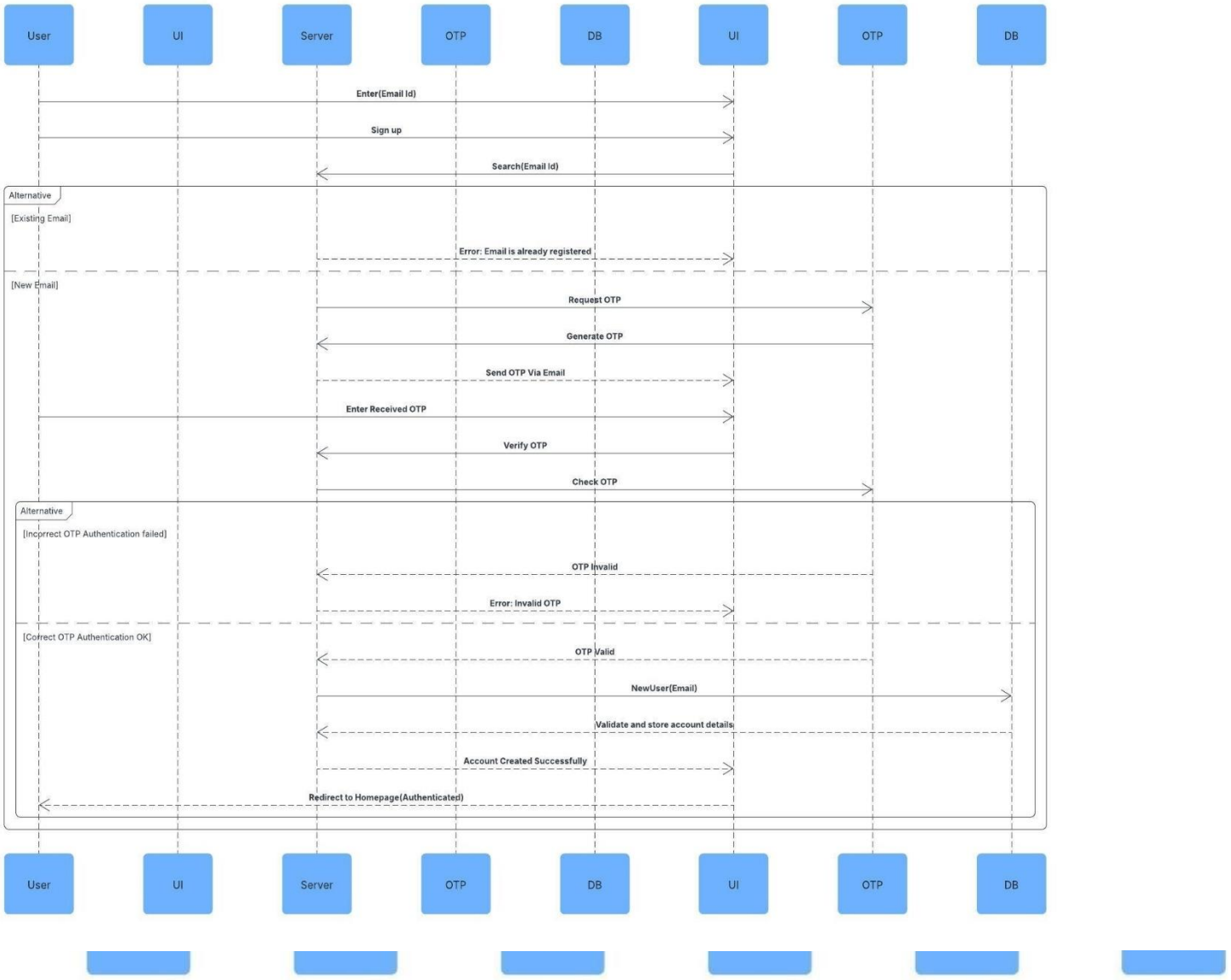


## 3.3 Sequence Diagrams

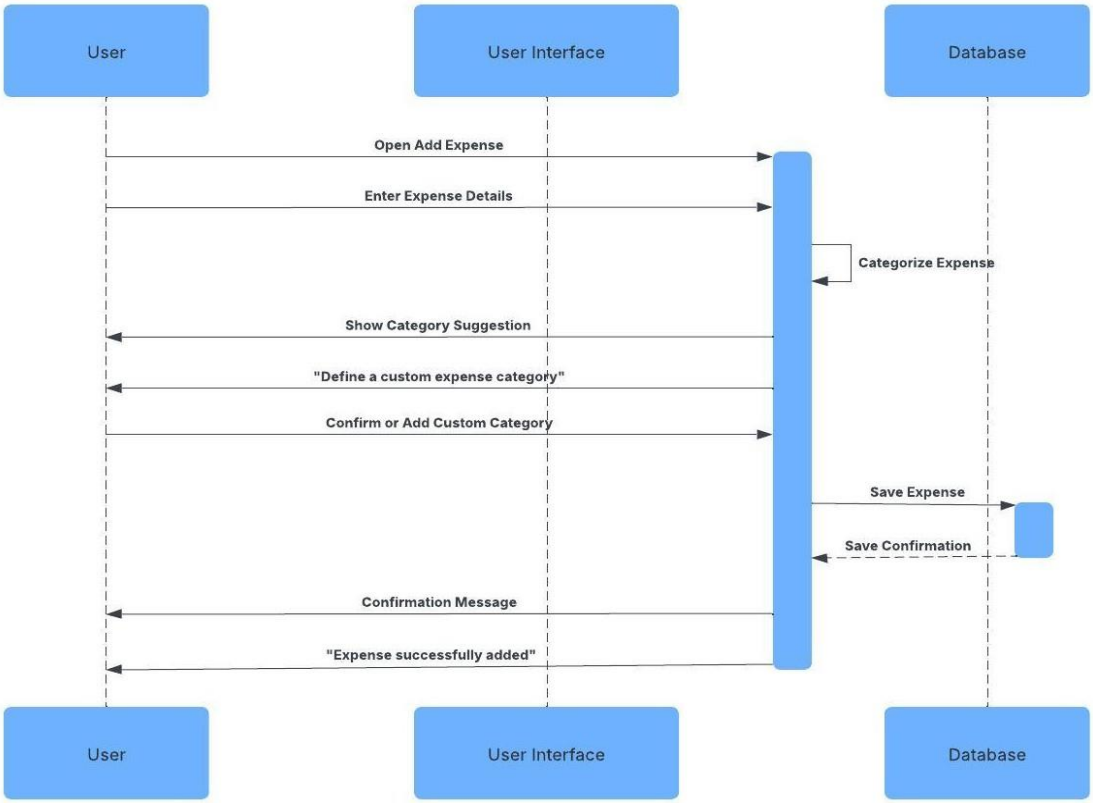
### 1. User Login



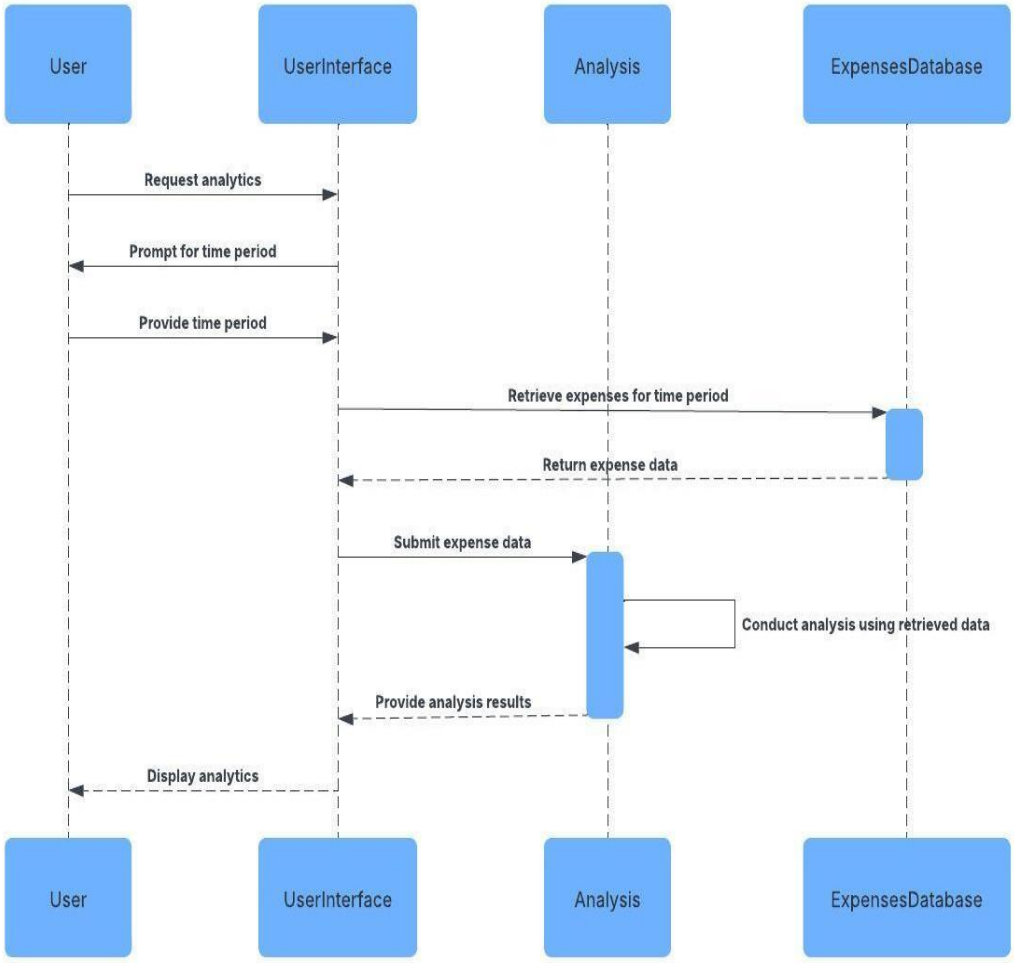
## 2. User Registration and Authentication



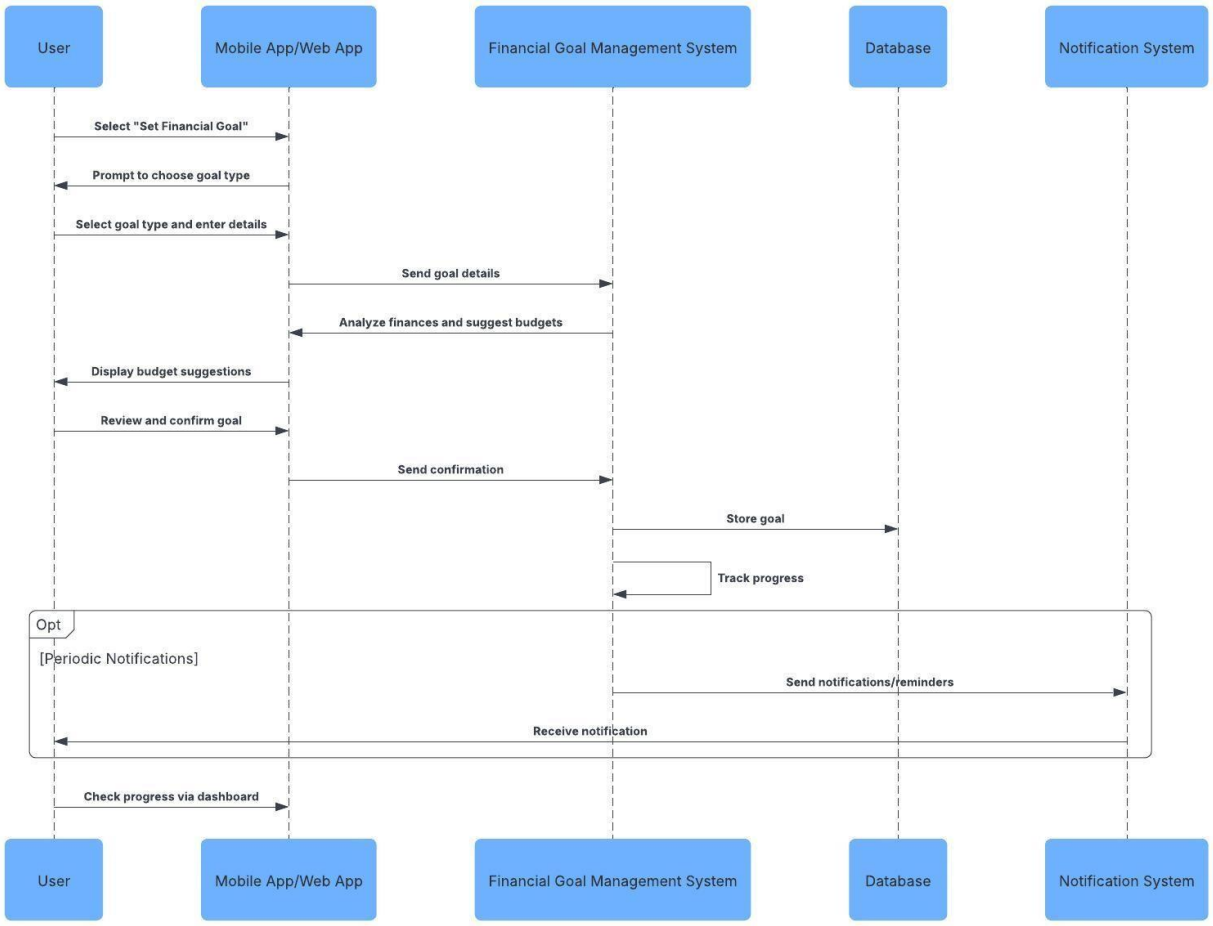
## 3. Expense Entry



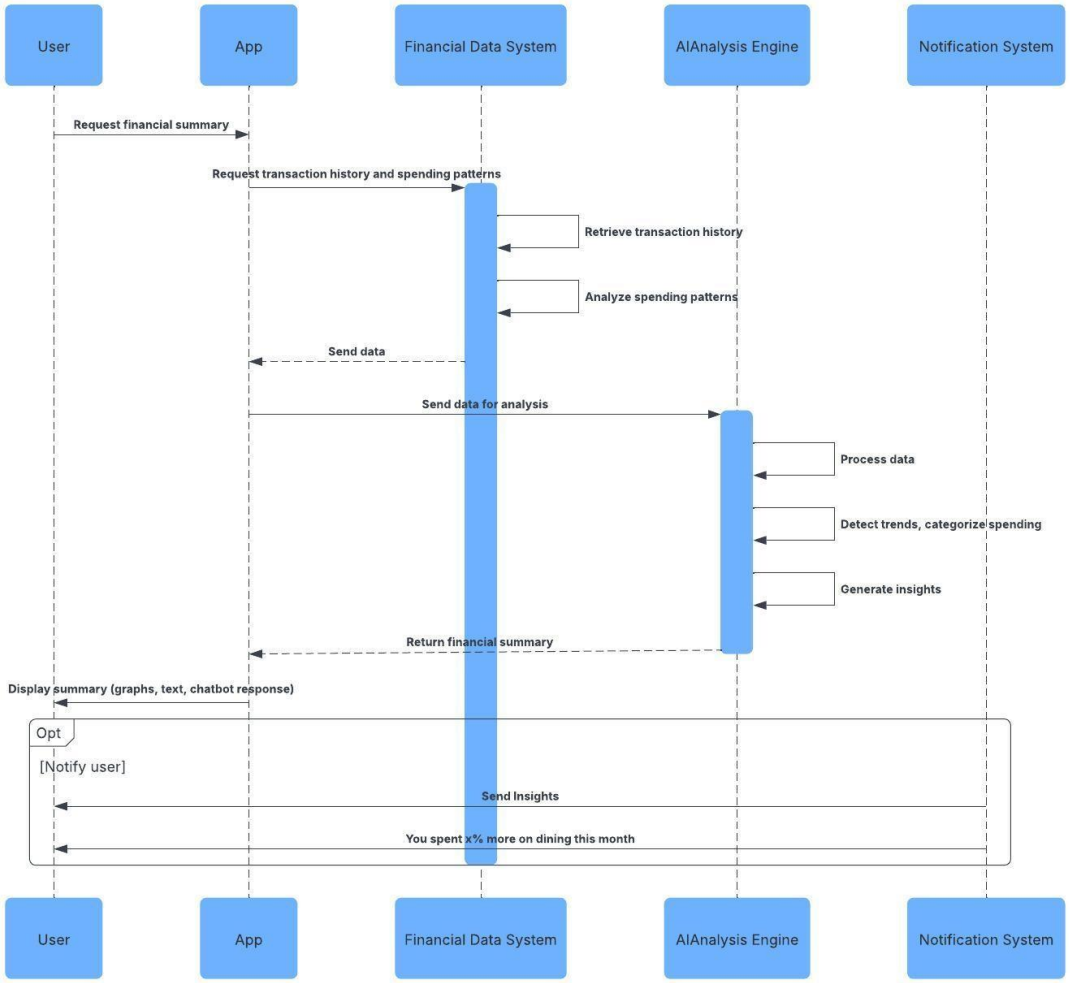
## 4. Spending Behavior Analysis:



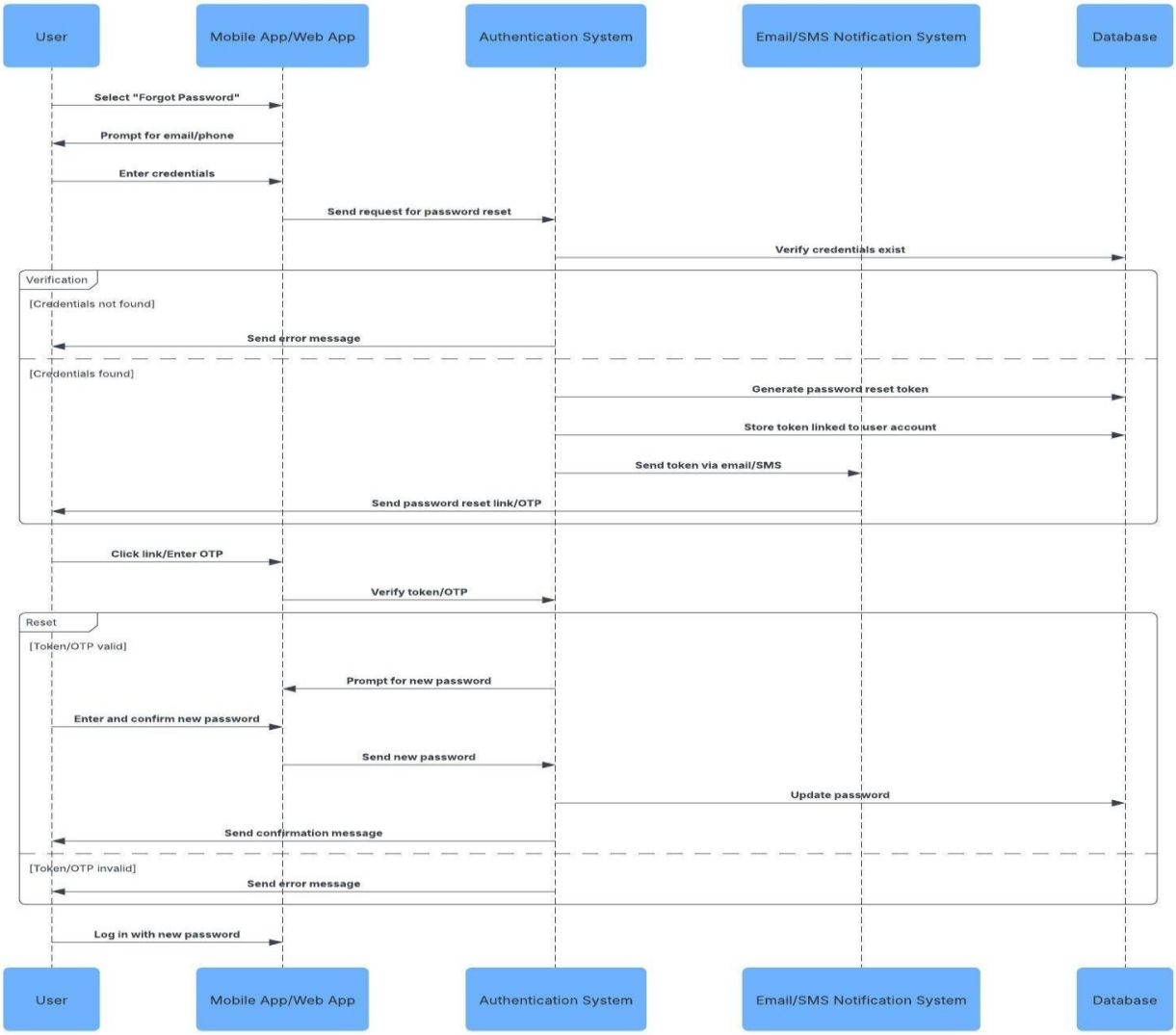
## 5. Setting Financial Goals



## 6. AI-Driven Financial Insights:

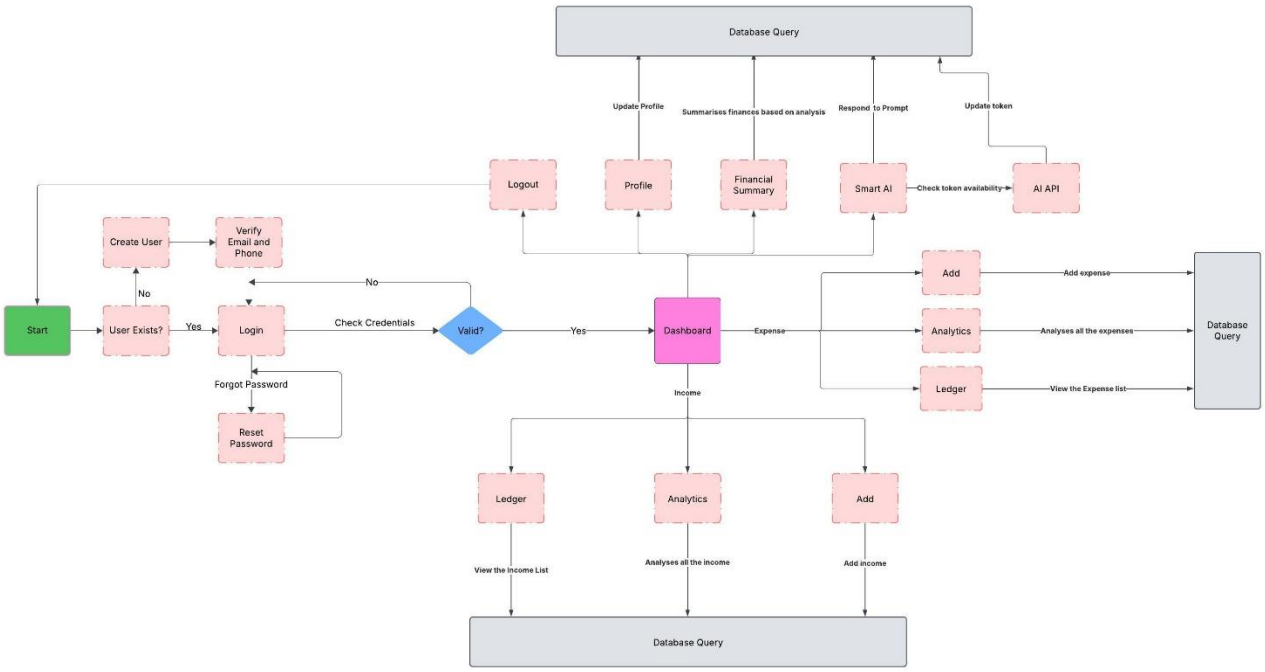


## 7. Password Reset and Account Reclaim:

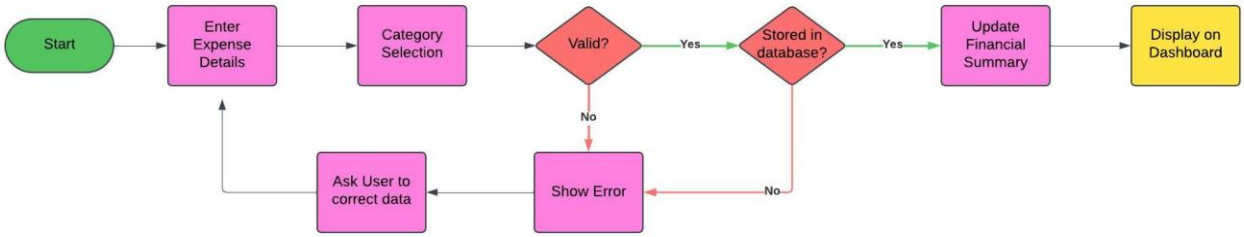


# 3.4 State Diagrams

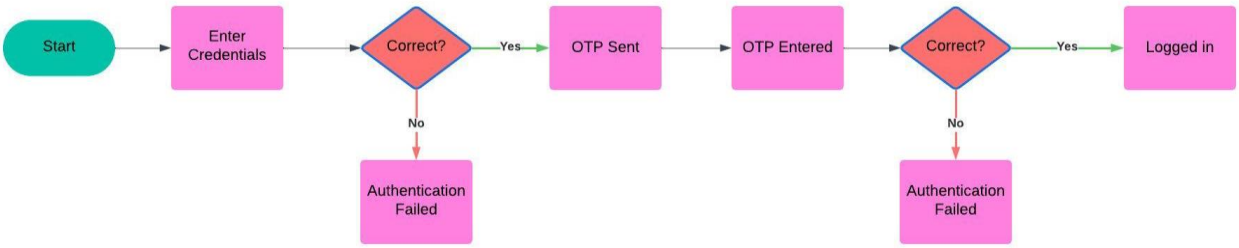
## Overall State Diagram



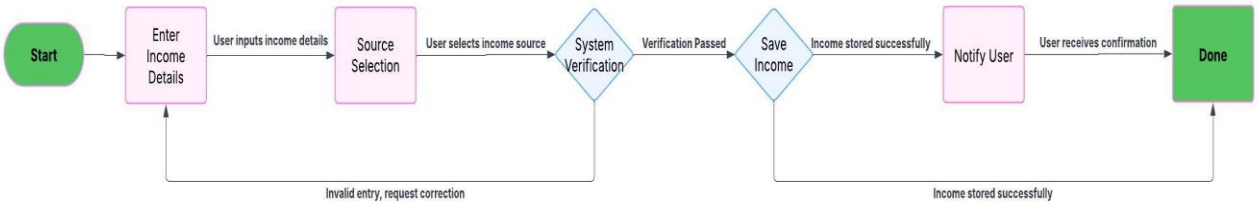

## 1. User Authentication



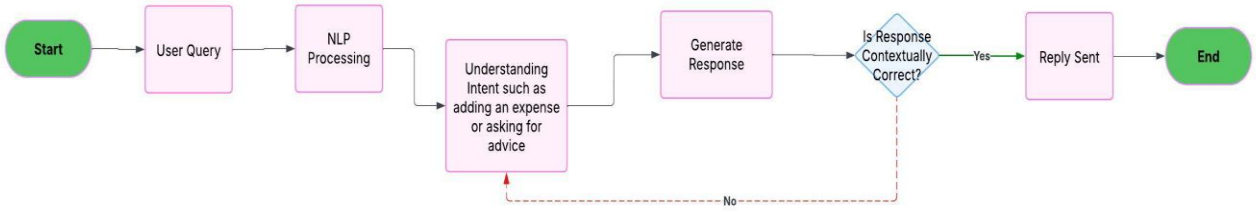

## 2. Expense Management



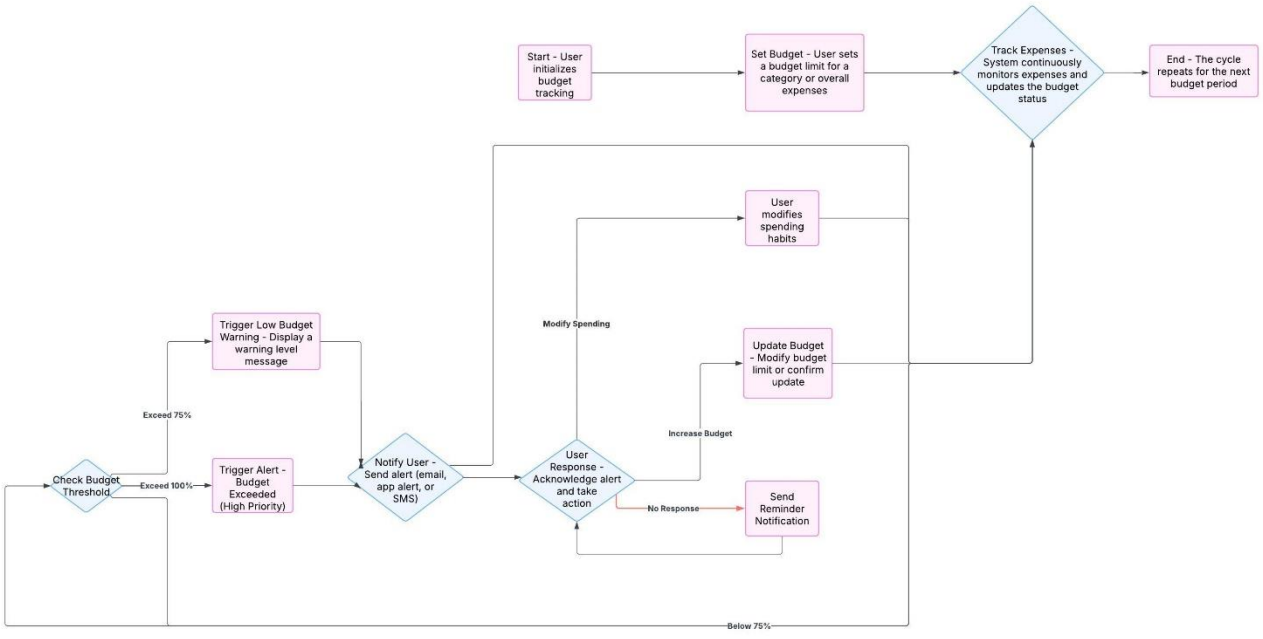
## 3. Income Management



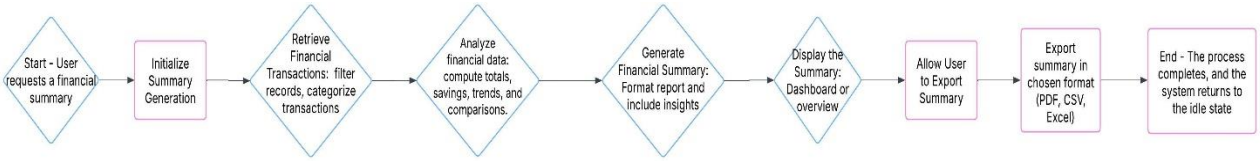

## 4. AI Chat Assistant Interaction



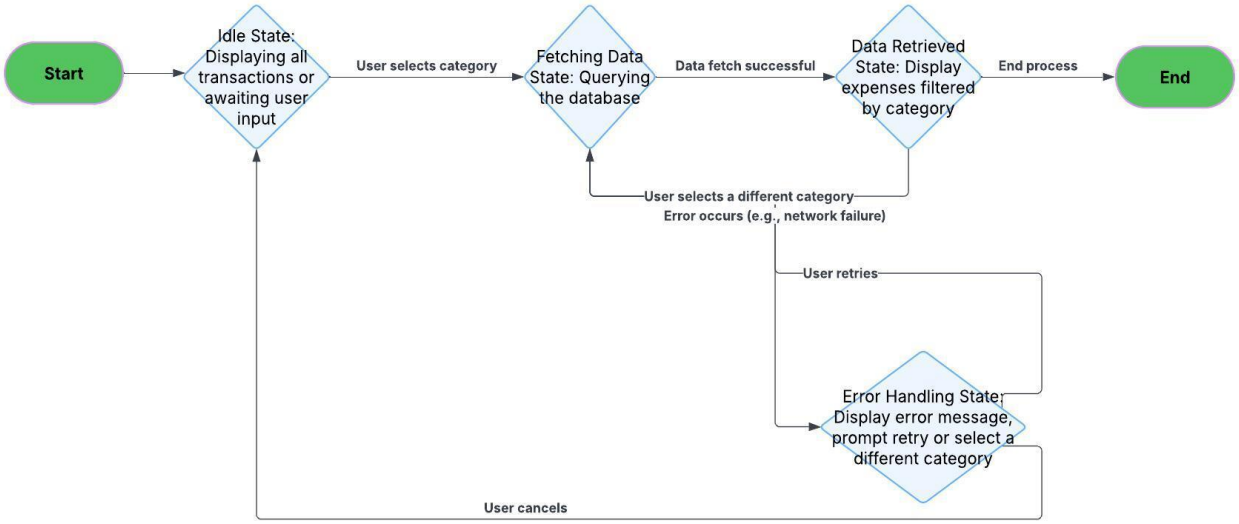

## 5. Budget Tracking & Alerts



## 6. Financial Summary Generation

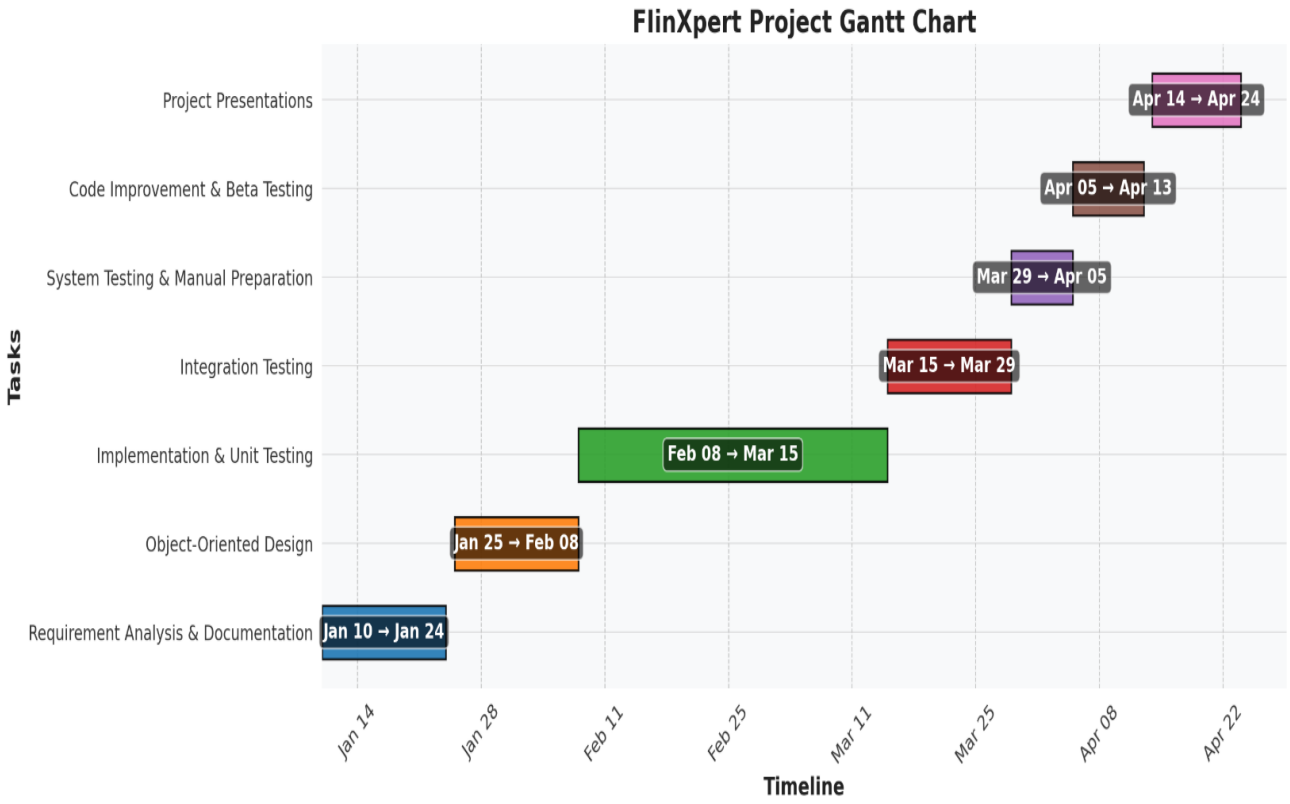



## 7. Ledger



# 4. Project Plan

## *Gantt Chart:*



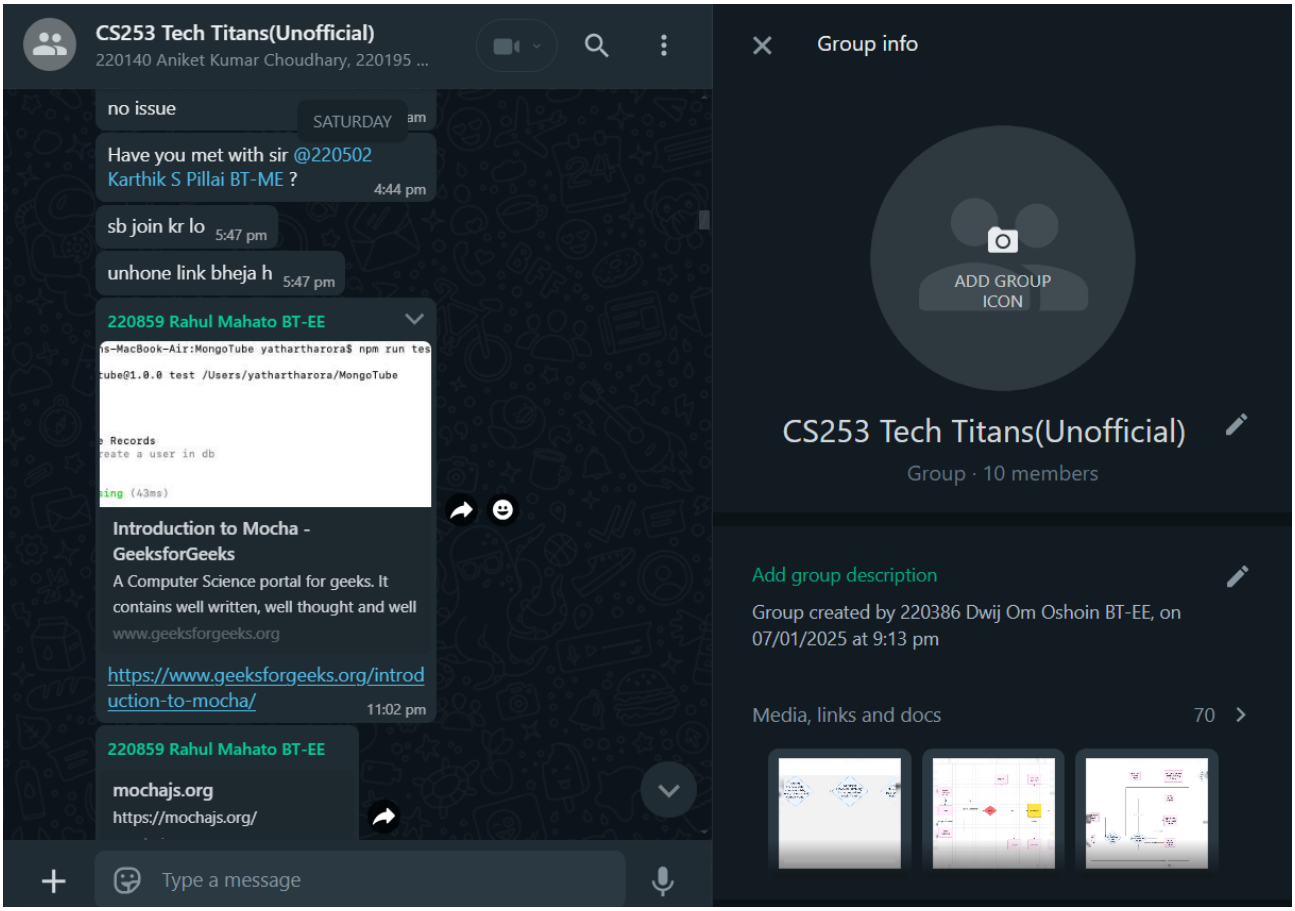

## *Team Members and Tasks*

| MEMBERS | TASK |
|---|---|
| DWIJ OM OSHOIN | FULL STACK IMPLEMENTATION, CODE IMPROVEMENT, ALPHA TESTING |
| KARTHIK S PILLAI | FULL STACK IMPLEMENTATION, CODE IMPROVEMENT, UNIT TESTING |
| SUMIT KUMAR | BACKEND DEVELOPMENT, CODE IMPROVEMENT, ALPHA TESTING |
| ANIKET KUMAR CHOUDHARY | FULL STACK IMPLEMENTATION, CODE IMPROVEMENT, UNIT TESTING |
| DEVANK SARAN PRAJAPATI | FULL STACK IMPLEMENTATION, SYSTEM TESTING, BETA TESTING |
| RAHUL MAHATO | BACKEND DEVELOPMENT, INTEGRATION TESTING, ALPHA TESTING |
| AYUSH YADAV | FRONTEND DEVELOPMENT, UNIT TESTING, ADDRESSING FEEDBACK |

| | |
|---|---|
| ARCHIT ATREY | BACKEND DEVELOPMENT, SYSTEM TESTING, ADDRESSING FEEDBACK |
| MADHAV KHETAN | BACKEND DEVELOPMENT, INTEGRATION TESTING, CODE IMPROVEMENT |
| PIYUSH MEENA | FRONTEND DEVELOPMENT, MANUAL FOR BETA TESTING, BETA TESTING |

### *Communication:*

*For the communication purpose we used a WhatsApp group. We also held offline meets at least twice a week, in which everybody was present, to ensure effective collaboration. After every meeting, the agenda and plan for the next meet used to be subsequently decided along with the division of work amongst team members.*



# Appendix A - Group Log

| Date | Timing | Duration | Agenda |
|---|---|---|---|
| 01/02/2025 | 20:00-23:00 | 3 hours | • Distribution of tasks for design document<br>• Divided into 4 teams<br>• Distributed roles of backend, frontend, and databasing for future |
| 03/02/2025 | 15:00-18:00 | 3 hours | • Explained approach of respective work to each other.<br>• Discussed doubts among team members. |
| 05/02/2025 | 19:00-21:00 | 2 hours | • Reviewed the work done by the four teams and suggested improvements.<br>• Clarified any irregularities in different tasks. |
| 06/02/2025 | 17:00-19:00 | 2 hrs | • Discussed project plan among team members. |
| 07/02/2025 | 17:30-19:30 | 2 hrs | • Finalised the document. |